MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ - UTFPR
CAMPUS CORNÉLIO PROCÓPIO
GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

Daniel Gonçalves da Silva
Lucas Lopes Dias da Silva
Gabriela dos Reis Bueno

RELATÓRIO TÉCNICO

N - Rainhas

CORNÉLIO PROCÓPIO

2018

Lista de Figuras

Resumo

O presente trabalho apresenta uma forma sobre a resolução do problema das N-Rainhas, proposto por Max Bazzel em 1848, de forma recursiva sob o uso de *backtracking*, bem como a introdução para funções de inteligência artificial de busca.

Palavras-chave: N-Rainhas, *backtracking*.

Sumário

1. INTRODUÇÃO

O problema das N-Rainhas, é um problema muito recorrente na literatura computacional e operacional, por tratar-se de um assunto que vincula diversas estruturas de dados, bem como conceitos introdutórios para a inteligência artificial - *com o algoritmo de backtracking.*

O problema consiste em posicionar enésimas rainhas (N) em um tabuleiro matricial de xadrez de NxN, e encontrar uma forma em que elas não se ataquem ou reduzam o número de colisões geradas. Lembrando que uma rainha pode caminhar tanto na vertical, quanto na horizontal ou diagonalmente. Como mostrado a seguir:

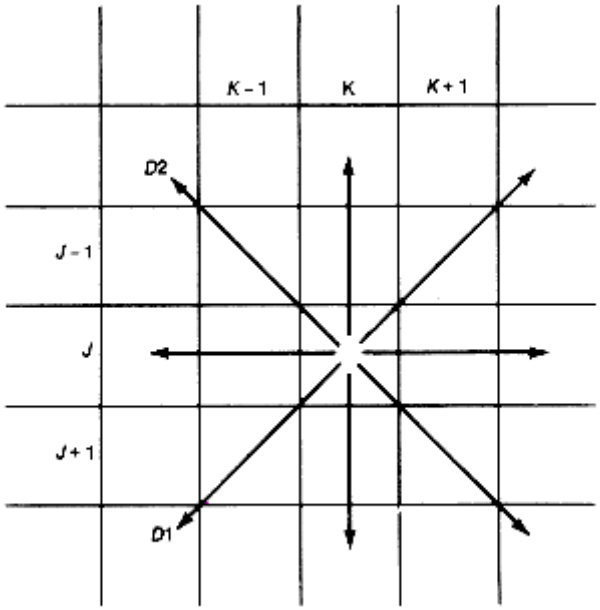


Figura 1: Exemplo de movimentação de uma rainha.
Fonte: *orion.lcg.ufrj.br/Dr.Dobbs/books/book2/algo02ae.htm*

2. OBJETIVOS

O trabalho tem como objetivo solucionar ou reduzir a quantidade de ataques gerados do problema das N-Rainhas em um tabuleiro matricial de 8 por 8, de uma matriz preenchida aleatoriamente com as N rainhas espalhadas pelo tabuleiro, de forma a rearranjar a disposição da matriz interpolando-as de forma que não se cruzem, gerando, assim, a menor quantidade possível de ataques ocasionados. Além de contabilizar a quantidade dos ataques realizados e contabilizar o tempo de execução do algoritmo.

3. N RAINHAS

O problema das n-rainhas a priori, fora esboçado por Max Bazzel, em 1848, um antigo jogador de xadrez, e somente em 1850 ganhara solução plausível. Grandes matemáticos e cientistas trabalharam nesse problema, como Carl Friedrich Gauss. [2]

Originalmente fora solucionado com 8 rainhas, posteriormente essa demonstração foi estendida para N-rainhas por Hoffman em 1969 com os avanços computacionais. [2]

O problema em si, consiste colocar um determinado número (N) maior ou igual a 4, de rainhas em um tabuleiro de xadrez de forma que fiquem dispostas na forma NxN (N-rainhas delimitam o tamanho do tabuleiro matricial), e que elas não se ataquem simultaneamente. Quando o tabuleiro de N-rainhas é gerado, as rainhas são posicionadas de forma aleatória sob o tabuleiro, e cabe ao algoritmo encontrar a melhor solução possível, onde reduzam o valor máximo do número de ataques gerados da disposição da matriz original (tabuleiro com as N-Rainhas). [2]

Para a resolução deste problema, utiliza-se, em particular, o algoritmo de “backtracking”, e algoritmos genéricos que possibilitam o modo de medir a eficiência na resolução. [2]

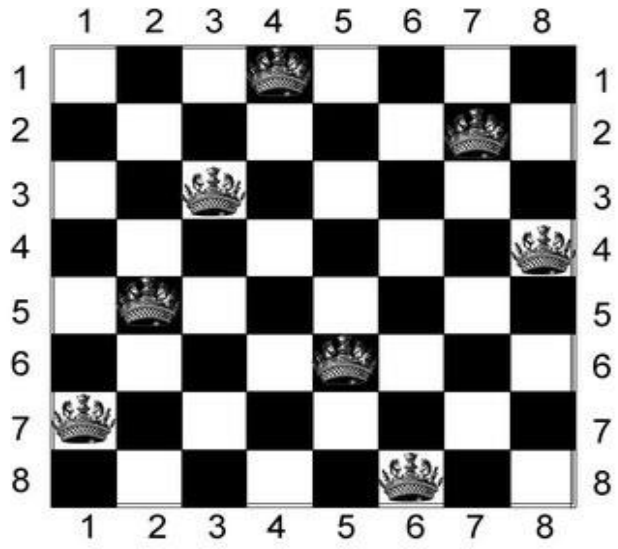


Figura 2: Uma das possíveis soluções para o problema das n-rainhas.

Fonte: *https://bit.ly/2BAkbP9*

3.1 BACKTRACKING

Existem inúmeras formas de abordagem para a resolução desse problema, desde funções estatísticas, funções por método de “força bruta” (ou enumeração exaustiva), funções recursivas e métodos heurísticos na angariação dos dados, mas para o projeto proposto fez-se uso do backtracking, um refinamento do método de “força bruta”. [2]

Backtracking refere-se a um tipo de algoritmo para encontrar todas (ou algumas) soluções de um problema computacional, que incrementalmente constrói possíveis soluções e abandona uma candidata parcialmente construída tão logo quanto for possível determinar que ela não pode gerar uma solução válida. [2]

Backtracking pode ser aplicado para problemas que admitem o conceito de “solução candidata parcial” e que exista um teste relativamente rápido para verificar se uma candidata parcial pode ser completada como uma solução válida. Quando aplicável, backtracking é frequentemente muito mais rápido que algoritmos de enumeração exaustiva (força bruta), já que ele pode eliminar um grande número de soluções inválidas com um único teste. [2]

Enquanto algoritmos de força bruta geram todas as possíveis soluções e só depois verificam se elas são válidas, backtracking só gera soluções válidas. [2]

Há diversas abordagens para a resolução do problema das N-Rainhas, entre as quais:

- “Força bruta” [3] – uma solução possível é, obviamente, testar todas as combinações possíveis de N rainhas em N2 posições. Mesmo para um valor de N igual a 10, as combinações possíveis são superiores a 1013. Uma simplificação possível é considerar que, após posicionar uma rainha, o tabuleiro restante fica uma dimensão mais pequeno (retirando a coluna e linha da rainha escolhida).

  Deste modo, as combinações possíveis são reduzidas a N!, o que representa ainda um grande esforço computacional (10! > 3x106).

- Abordagem recursiva com “backtracking” [2][3] –colocam-se as rainhas, uma de cada vez e uma por coluna, verificando se cada uma não é atacada. Se for, move-se a rainha para a linha seguinte. Caso já não existam posições seguras, volta-se à rainha da coluna anterior e procura-se nova posição. Se a rainha estiver segura, continua-se o processo na primeira linha da coluna seguinte. Este é, provavelmente, o algoritmo mais popular, já que tem uma eficiência aceitável e é relativamente simples de implementar.

- Algoritmo permutativo [3] – colocam-se as N rainhas e efetuam-se sucessivas permutações, verificando se a sua disposição é (ou não) uma solução válida. Tem semelhanças com o primeiro método, embora faça a restrição de uma rainha por coluna. Uma variante é colocar inicialmente as rainhas em posições com conflitos mínimos (mas possíveis) e, posteriormente, fazer permutações sucessivas até encontrar uma solução. desdobramento do projeto proposto.

A escolha para a realização do projeto, fora utilizar o método de *backtracking*.

4. ABORDAGENS PARA SOLUCIONAMENTO DO PROBLEMA

Nossa estratégia de busca resultou de uma análise sobre o tempo de resposta sobre o trabalho proposto, como N deveria ser 8, optamos pela busca cega, com o algoritmo de busca em profundidade, pelo uso de backtracking. [1]

O método de busca cega funciona bem para Ns menores, como é o caso do projeto proposto. Caso o N necessário fosse muito maior, os métodos de busca heurístico fariam uma melhor escolha. [1]

4.1 BUSCA CEGA

O algoritmo de busca cega inicia a com a matriz tabular vazia e vai inserindo a solução coluna a coluna, da esquerda para a direita, inserindo a solução ou a melhor posição na coluna vazia mais à esquerda. [1]

Busca em profundidade: Cada sequência de busca, o topo das colunas é analisado e caso seja um estado aceitável (referência ao confronto das rainhas), o próximo estado não visitado a partir de uma expansão deste estado é inserida na pilha e, caso não contenham mias expansões (sem confronto) possíveis, ele é removido da pilha e o estado anterior analisado por backtracking. Esse algoritmo funciona bem para N não tão grandes, e apresenta uma melhora em relação à busca em largura, apesar de manter o caráter exponencial. Para Ns > 24 o tempo acaba não tornando-o razoável, mas nosso N será de 8, optamos por essa escolha. [1]

5. FUNÇÕES PARA SOLUCIONAMENTO DO PROBLEMA

A função resolveNRUtil(), através do método de backtracking resolve o problema das N-Rainhas.

```
bool resolveNRUtil(int **mat, int col, int nu){
   int N = nu;
   if (col >= N)
         return true;
   int i;
   for (i = 0; i < N; i++){
      if ( colocaQueen(mat, i, col, N) ) {
                  mat[i][col] = 2;
         if ( resolveNRUtil(mat, col + 1, N) )
            return true;
                  mat[i][col] = 0; // BACKTRACK
      }
   }
   return false;
}
```

Esta função resolve o problema N das Rainhas usando retrocesso. Utiliza principalmente resolveNQUtil() para resolva o problema. Ele retorna falso se rainhas não pode ser colocado, caso contrário, retornar *true* e imprime colocação de rainhas na forma de 2. Note que pode haver mais de uma solução, esta função imprime uma das soluções viáveis.

```
bool resolveNR(int **matz, int n){
        int **mat = matz;
        int nu=n;
        liberaQueen(mat,nu);//
        if (resolveNRUtil(mat, 0, nu) == false ) {
        printf("Solução não existe");
                return false;
        }
   imprimeSol(mat,nu);
   return true;
}
```

6. VISUALIZAÇÃO

O número 2 é atribuído as Rainhas, e num primeiro instante fora gerado um tabuleiro matricial de NxN com N=8 contendo 6 confrontos.

```
informe numero de rainhas: 8

Tabuleiro inicial com numero de casas igual N rainhas: 8 x 8

0   0   0   0   0   0   0   0
0   0   0   0   0   0   2   0
2   0   0   0   0   0   0   0
0   0   0   0   0   2   0   2
0   0   0   2   0   0   0   0
0   0   0   0   0   0   0   0
0   0   2   0   2   0   0   0
0   2   0   0   0   0   0   0
```

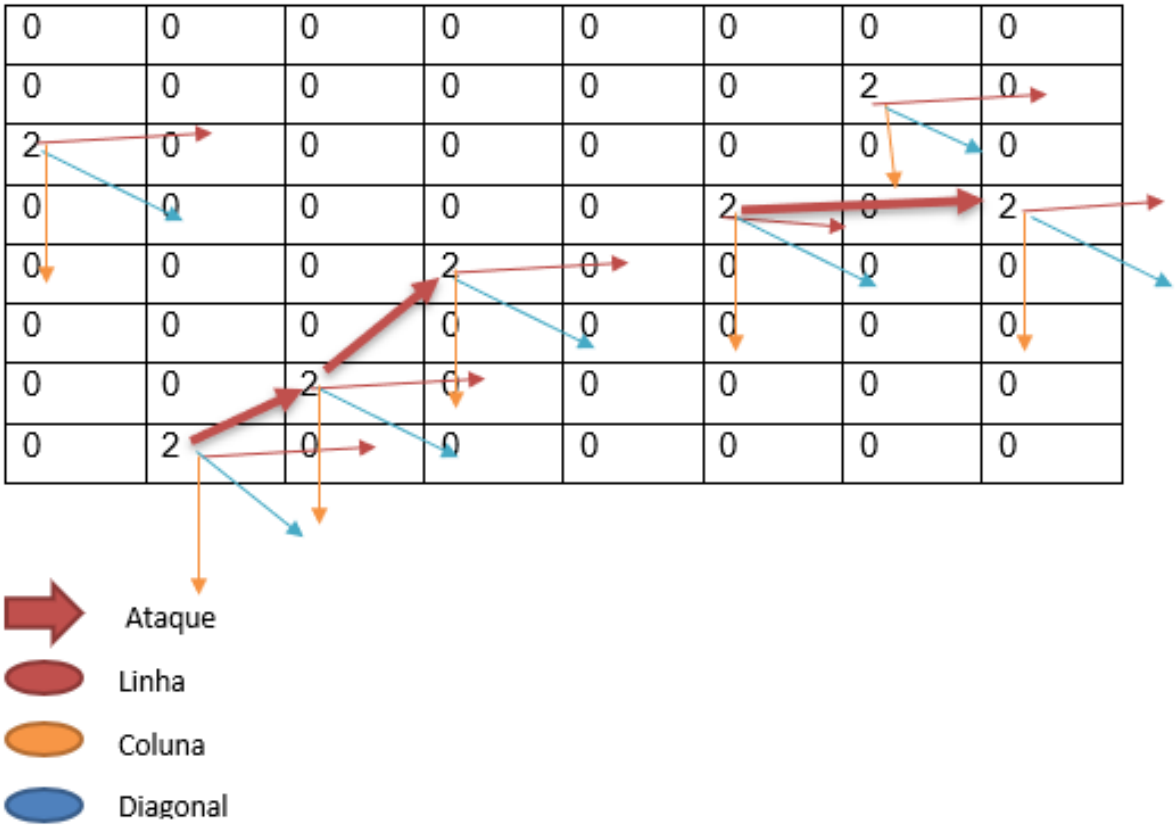


Figura 3: Disposição matricial gerada para o problema das N-rainhas.

Nessa segunda etapa nosso algoritmo conseguiu reduzir os confrontos em 0, rearranjando o tabuleiro da seguinte forma:

```
Informe numero de rainhas: 8

Tabuleiro inicial com numero de casas igual N rainhas: 8 x 8

0   0   0   0   0   0   0   0
0   0   0   0   0   0   2   0
2   0   0   0   0   0   0   0
0   0   0   0   0   2   0   1
0   0   0   2   0   0   0   0
0   0   0   0   0   0   0   0
0   0   2   0   2   0   0   0
0   2   0   0   0   0   0   0
Tabuleiro final com a solucao para 8 rainhas que nao se atacam.

2   0   0   0   0   0   0   0
0   0   0   0   0   0   2   0
0   0   0   0   2   0   0   0
0   0   0   0   0   0   0   2
0   2   0   0   0   0   0   0
0   0   0   2   0   0   0   0
0   0   0   0   0   2   0   0
0   0   2   0   0   0   0   0

--------------------------------
Process exited after 2.725 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

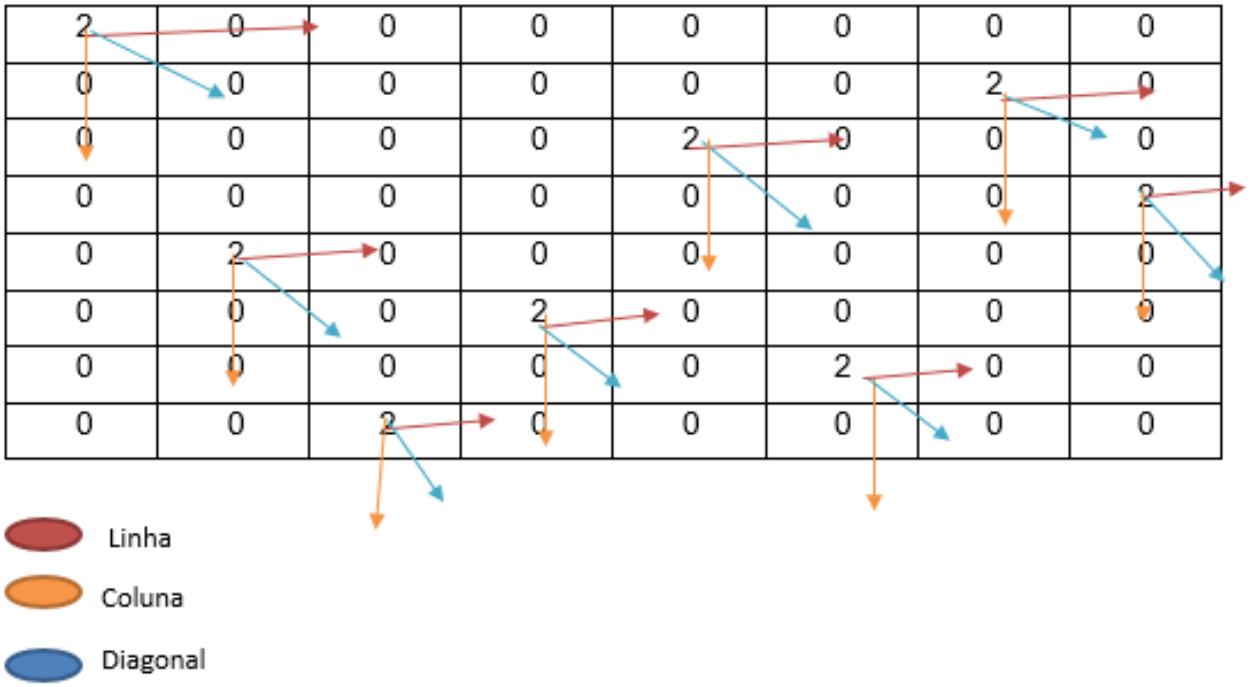
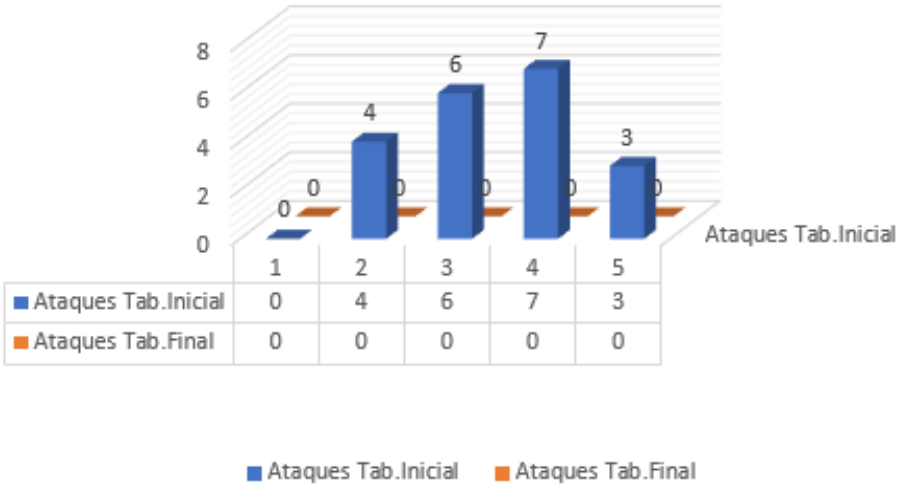


Figura 4: Solução baseada no algoritmo do grupo.

Com algumas execuções do algoritmo, chegamos na seguinte informação:



7. CONCLUSÃO

Concluímos, portanto, que o algoritmo gerado pelo grupo funciona bem para N menores que 30, reduzindo o número de ataques numa parcela grande de execuções do programa e encontrando uma solução plausível.

O entendimento sobre recursividade e backtracking fora de vital importância, sem discutir o know-how operacional de alguns sistemas inteligentes propostos na disciplina futura de Inteligência artificial.

8. REFERÊNCIAS BIBLIOGRÁFICAS

[1] WITT, Thiago Moura. O PROBLEMA DAS N-RAINHAS. Introdução à Inteligência Artificial. IME-USP. 2003. Acesso em: 15 de out. 2018.

[2] ALGORITMOS E PROGRAMAÇÃO DE COMPUTADORES. UNICAMP. 2013. Acesso em 20/11/2018. http://www.ic.unicamp.br/~zanoni/mc102/2013-1s/aulas/aula22.pdf

[3] FERNANDES, Marco. ALHO, Miguel. Solução eficiente para a resolução do Problema das N-Rainhas. REVISTA DO DETUA, VOL. 4, No 2, JANEIRO 2004.